BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pendukung Keputusan

Menurut Hilyah M. (2012) Konsep Sistem Pendukung Keputusan pertama kali diperkenalkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision System* (Sprague, 1982). Konsep pendukung keputusan ditandai dengan sistem interaktif berbasis komputer yang membantu pengambil keputusan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tidak terstruktur. Pada dasarnya SPK diracang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam prosen pengambilan keputusan, sampai mengevaluasi pemilihan alternatif.

Menurut Iskandar Z. Nasibu Vol. 2 No. 5 (2009) Sistem Pendukung Keputusan (*Decision Support System/DSS*) adalah sebuah sistem yang memberikan dukungan kepada seorang manajer, atau kepada sekelompok manajer yang relatif yang bekerja sebagai tim pemecah masalah, dalam memecahkan masalah semi terstruktur dengan memberikan informasi atau saran mengenai keputusan tertentu. Informasi tersebut dapat diberikan dalam bentuk laporan berkala, laporan khusus, maupun output dalam model matematis. Model tersebut juga mempunyai kemampuan untuk memberikan saran dalam tingkat yang bervariasi.

Menurut Ahmad Khaidir (2014), Pengambilan keputusan merupakan proses pemilihan alternative tindakan untuk mencapai tujuan atau sasaran tertentu. Pengambilan keputusan dilakukan dengan pendekatan sistematis terhadap permasalahan melalui proses pengumpulan data menjadi informasi serta ditambah dengan faktor - faktor yang perlu dipertimbangkan dalam pengambilan keputusan.

Menurut Keen dan Scoot Morton : "Sistem Pendukung Keputusan merupakan penggabungan sumber-sumber kecerdasan individu dengan kemampuan komponen untuk memperbaiki kualitas keputusan. Sistem Pendukung Keputusan juga merupakan sistem informasi berbasis komputer untuk manajemen pengambilan keputusan yang menangani masalah-masalah semi struktur ".

Dengan pengertian diatas dapat dijelaskan bahwa sistem pendukung keputusan bukan merupakan alat pengambilan keputusan, melainkan merupakan sistem yang membantu pengambil keputusan dengan melengkapi mereka dengan informasi dari data yang telah diolah dengan relevan dan diperlukan untuk membuat keputusan tentang suatu masalah dengan lebih cepat dan akurat. Sehingga sistem ini tidak dimaksudkan untuk menggantikan pengambilan keputusan dalam proses pembuatan keputusan.

Alter (2002:38) mendefinisikan Sistem pendukung keputusan atau *Decision Support Systems* (DSS) adalah sistem informasi interkatif yang menyediakan informasi, pemodelan, dan pemanipulasian data yang digunakan untuk membantu pengambilan keputusan pada situasi yang semiterstruktur dan situasi yang tidak terstruktur dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat. Konsep DSS dikemukaakan pertama kali oleh Scoot Morton pada tahun 1971 (Turban, McLean, dan Wetherbe, 1999).

Beliau mendefinisikan cikal bakal DSS tersebut sebagai : "*system* berbasis computer yang interaktif, yang membantu pengambil keputusan dengan menggunakan data dan model untuk memecahkan persoalan-persoalan tak terstruktur"

II.1.1. Tujuan Sistem Pendukung Keputusan

Dalam Heny Pratiwi (2014), Menurut Turban (2005), tujuan sistem pendukung keputusan yaitu :

1. Membantu manajer dalam mengambil keputusan atas masalah semi terstuktur.
2. Memberikan dukungan atas pertimbangan manajer dan bukan dimaksudkan untuk menggantikan manajer.
3. Meningkatkan efektivitas keputusan yang diambil manajer lebih dari pada perbaikan efesiensinya.
4. Kecepatan komputasi. Komputer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas. Membangun satu kelompok pengambilan keputusan, terutama oleh para pakar, akan meningkatkan biaya. Pendukung berupa perangkat terkomputerisasi dapat mengurangi kelompok dan memungkinkan anggotanya untuk berada diberbagai lokasi yang berbeda-beda.
6. Meningkatkan kualitas. Komputer dapat meningkatkan kualitas keputusan yang dibuat.

7. Berdaya asing. Manajemen dan pemberdayaan sumber daya perusahaan.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan.

II.1.2. Tahap-Tahap Pengambilan Keputusan

Dalam Hilyah M. (2012), Menurut Herbert A. Simon (Kadarsyah Suryadi dan Ali Ramdhani, 2002, 15-16) model yang menggambarkan proses pengambilan keputusan. Proses Pengambilan Keputusan melibatkan 4 tahapan, yaitu:

1. Tahap *Intelligence*

   Dalam tahap merupakan proses penelusuran dan pendeteksian dari lingkup problematika serta proses pengenalan masalah. Data masukan diperoleh, diproses, dan diuji dalam rangka mengidentifikasi masalah.

2. Tahap *Design*

   Dalam tahap ini merupakan proses menemukan, mengembangkan dan menganalisis alternatif tindakan yang bisa dilakukan.

3. Tahap *Choice*

   Dalam tahap ini dilakukan proses pemilihan diantara berbagai alternatif tindakan yang mungkin dijalankan.

4. Tahap *Implementation*

   Dalam tahap ini pengambil keputusan menjalankan rangkaian aksi pemecahan yang dipilih di tahap *choice.* Implementasi yang sukses ditandai dengan terjawabnya masalah yang dihadapi, sementara kegagalan ditandai dengan tetap adanya masalah yang sedang dicoba untuk diatasi.

II.1.3. Komponen Sistem Pendukung Keputusan

Dalam Nila Susanti dan Sri Winiarti (2013) Sistem Pendukung Keputusan terdiri atas empat komponen utama, yaitu:

1. Subsistem manajemen data berfungsi sebagai memasukkan suatu *database* yang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut yang disebut sistem manajemen *database* (DBMS). *Knowledge Base* berisi semua fakta, ide, hubungan dan interaksi suatu domain tertentu.
2. Subsistem manajemen basis pengetahuan bertugas untuk mendukung semua subsistem lain atau bertindak sebagai suatu komponen independen. Ia memberikan intelegensi untuk memperbesar pengetahuan pengambil keputusan.
3. Subsistem manajemen model merupakan paket perangkat lunak yang memasukkan model keuangan statistik, ilmu manajemen atau model kuantitatif lainnya yang memberikan kapabilitas analitik dan manajemen perangkat lunak yang tepat.
4. Subsistem antarmuka pengguna (dialog) untuk mengimplementasikan sistem kedalam program aplikasi sehinggga pengguna atau pemakai dapat berkomunikasi dengan sistem yang dirancang.

Dalam Heny Pratiwi (2014), Menurut Sudiyantoro (2005), komponen dalam sistem pendukung keputusan meliputi 8 bagian yaitu:

1. Perangkat Keras

Perangkat keras ini akan terhubung dengan komputer lain menggunakan sistem jaringan, sehingga memudahkan dalam pengambilan data pada organisasi tersebut.

2. Perangkat Lunak

Perangkat lunak sistem pendukung keputusan sering disebut juga dengan *DSS Generator,* berisi modul-modul untuk database, model dan *dialogue management*.

3. Sumber Data

Database sistem pendukung keputusan berisi data dan informasi yang diambil dari dari data organisasi, ekternal dan data manajer secara individu.

4. Sumber Model

Model ini berisi kumpulan model matematika dan teknik analisis yang disimpan dalam program dan berkas yang berbeda-beda. Komponen dari model ini dapat di kombinasikan dengan perangkat lunak tertentu untuk mendukung sebuah keputusan yang akan diambil.

5. Sumber Daya Manusia

Sistem pendukung keputusan dapat digunakan oleh para manajer dan staf khusus untuk membuat keputusan ini juga dapat dikembangkan oleh penggunanya sesuai dengan keperluan para pengguna tersebut.

6. Model Sisem Pendukung Keputusan

Model merupakan komponen yang penting dalam SPK. Model memilii pengertian yang berarti memisahkan dari dunia nyata dengan melukiskan komponen utama dan menghubungkannya dengan sistem dan kejadian lainnnya.

7. Lembar Kerja Elektronik

Lembar kerja elektronik memudahkan pengguna membuat model dengan cara mengisi data dan menghubungkannya sesuai dengan format yang telah disediakan. Pengguna dapat melakukan beberapa perubahan dan mengevaluasi secara tampilan grafik.

8. Sistem Pendukung Keputusan Kelompok

Merupakan suatu sistem berbasis komputer yang mendukung kelompok-kelompok orang yang terlibat dalam suatu tugas atau tujuan bersama dan menyediakan tampilan antarmuka pada sat lingkungan yang digunakan bersama.

II.1.4. Kriteria Sistem Pendukung Keputusan

Dalam Dita Monita (2013) Sistem pendukung keputusan dirancang secara khusus untuk mendukung seseorang yang harus mengambil keputusan-keputusan tertentu [1]. Berikut ini beberapa criteria sistem pendukung keputusan, yaitu:

1. Interaktif

Sistem pendukung keputusan memiliki *user interface* yang komunikatif sehingga pemakai dapat melakukan akses secara cepat ke data dan memperoleh informasiyang dibutuhkan.

2. Fleksibel

Sistem pendukung keputusan memiliki sebanyak mungkin variable masukan, kemampuan untuk mengolah dan memberikan keluaran yang menyajikan alternatif-alternatif keputusan kepada pemakai.

3. Data Kualitas

Sistem pendukung keputusan memiliki kemampuan untuk menerima data kualitas yang dikuantitaskan yang sifatnya subyektif dari pemakainya, sebagai data masukan untuk pengolahan data. Misalnya terhadap kecantikan yang bersifat kualitas, dapat dikuantitaskan dengan pemberian bobot nilai seperti 75 atau 90.

4. Prosedur Pakar

Sistem pendukung keputusan mengandung suatu prosedur yang dirancang berdasarkan rumusan formal atau juga berupa prosedur kepakaran seseorang atau kelompok dalam menyelesaikan suatu bidang masalah dengan fenomena tertentu.

II.1.5. Struktur Keputusan dalam Sistem Pendukung Keputusan

Dalam Jurnal Heny Pratiwi (2014), Menurut Kusrini (2007), keputusan yang diambil untuk menyelesaikan suatu masalah dilihat dari stuktur masalahnya terbagi menjadi tiga yaitu :

1. Keputusan terstuktur (*Structured Decision*)

   Keputusan yang dilakukan secara berulang-ulang dan bersifat rutin. Prosedur pengambilan keputusan sangat jelas. Keputusan tersebut dilakukan pada manajemen tingkat bawah (operasional). Misalnya, keputusan pemesanan barang dan keputusan penagihan piutang.

2. Keputusan semi terstuktur (*Semi Structured Decision*)

   Keputusan yang memiliki dua sifat. Sebagian keputusan bisa ditangani oleh komputer sedangkan yang lain tetap harus dilakukan oleh pengambil keputusan. Produser dalam pengambilan keputusan tersebut secara garis besar sudah ada, tetapi beberapa hal yang masih memerlukan kebijakan dari pengambil keputusan. Prosedur dalam pengambilan keputusan tersebut secara garis besar sudah ada, tetapi beberapa hal yang masih memerlukan kebijakan dari pengambilan keputusan. Biasanya keputusan semacam ini diambil oleh manajer tingkat menengah (taktikal). Contoh keputusan jenis ini adalah evaluasi kredit, penjadwalan produksi dan pengendalian persediaan.

3. Keputusan Tak Terstruktur (*Unstructured Decision*)

   Keputusan yang penanganannya rumit karena tidak selalu terjadi. Keputusan tersebut menurut pengalaman dan berbagai sumber yang

bersifat eksternal. Keputusan tersebut umumnya terjadi pada manajemen tingkat atas (strategis). Contonya adalah keputusan untuk pengembangan teknologi baru, keputusan bergabung dengan perusahaan lain dan perekrutan eksekutif.

II.2. Metode *Simple Additive Weighting*

Menurut Youllia Indrawaty; 2011:33. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada.

$$r_{ij} = \begin{cases} \dfrac{X_{ij}}{\text{Max } X_{ij}} & \text{Jika j atribut Keuntungan (}\textit{benefit}\text{)} \\ \dfrac{\text{Min } X_{ij}}{i} & \text{Jika j atribut biaya (}\textit{cost}\text{)} \end{cases}$$

$$X_{ij}$$

Keterangan :

Rij = Nilai rating kinerja normalisasi

Xij = Nilai atribut yang dimiliki dari setiap kriteria

Max xij = Nilai terbesar dari setiap kriteria

Min xij = Nilai terkecil dari setiap kriteria

Benefit = Nilai terbesar adalah terbaik

Cost = Nilai Terkecil adalah terbaik

Dimana rij adalah rating kinerja ternormalisasi dari alternatif Ai pada atribut Cj; i=1,2,.....,m dan j=1,2.....,n. Nilai *preferensi* untuk setiap alternatif (Vi) diberikan sebagai :

$$V_i = \sum_{J=1}^{n} w_j r_{ij}$$

Keterangan :

V_i = Rangking untuk setiap alternatif

W_j = Nilai bobot dari setiap kriteria

r_{ij} = Nilai rating kinerja ternormalisasi

Nilai Vi yang lebih besar mengindikasi bahwa alternatif Ai lebih terpilih.

Langkah – langkah dari metode SAW adalah :

1. Menentukan kriteria – kriteria yang akan dijadikan acuan dalam pengambilan keputusan, yaitu C.
2. Menentukan rating kecocokan setiap alternatif pada setiap kriteria.
3. Membuat matriks keputusan berdasarkan kriteria (C), kemudian melakukan normalisasi matriks berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan ataupun atribut biaya) sehingga diperoleh matriks ternormalisasi R.
4. Hasil akhir diperoleh dari proses peranking yaitu penjumlahan dari perkalian matriks ternormalisasi R dengan vector bobot sehingga

diperoleh nilai terbesar yang dipilih sebagai alternatif terbaik (A) sebagai solusi.

II.2.1 Kelebihan Metode Simple Additive Weighting (SAW)

Kelebihan dari model Simple Additive Weighting (SAW) dibandingkan dengan model pengambilan keputusan yang lain terletak pada kemampuannya untuk melakukan penilaian secara lebih tepat karena didasarkan pada nilai kriteria dan bobot preferensi yang sudah ditentukan, selain itu SAW juga dapat menyeleksi alternatif terbaik dari sejumlah alternatif yang ada karena adanya proses perankingan setelah menentukan nilai bobot untuk setiap atribut. (Darmastuti Destriyana, 2010)

II.3. Cadangan *Bad Debt*

Menurut Hendriksen (2010 : 269) Piutang tak tertagih *(Bad Debt)* adalah Piutang yang dinilai tidak lagi dapat ditagih dan ditargetkan untuk dihapuskan dari daftar piutang dalam proses yang dikenal sebagai "*write off*" (penghapusan piutang). Dalam bisnis, menjual barang dan jasa secara kredit adalah hal yang biasa, dengan penjualan kredit tersebut berarti sebagai penjual akan mencatatkan piutang. Tetapi ada kalanya piutang-piutang tersebut tidak dapat tertagih, bisa karena banyak alasan. Untuk itu manajemen biasanya akan membuat langkah antisipasi, salah satunya dengan membentuk cadangan *bad debt*. Secara umum cadangan *bad debt* merupakan suatu metode akuntansi yang mengacu pada prinsip akuntansi konservatisme atau kehati-hatian, dan sebagai alat bagi perusahaan untuk menaksir risiko atas kemungkinan tidak tertagihnya suatu potensi

pendapatan yaitu piutang. Dimana cadangan *bad debt* merupakan kompensasi tidak langsung yang diberikan kepada karyawan dalam rangka menumbuhkan kepuasan dan ketenangan kerja.

Cadangan *Bad Debt* merupakan Selisih dari tunggakkan yang bisa diturunkan dan salah satu komponen insentif yang diberikan oleh perusahaan atau organisasi kepada karyawannya. Dimana cadangan *bad debt* merupakan imbalan yang diberikan kepada pegawai yang mampu bekerja sedemikian rupa sehingga tingkat produktivitas yang berlaku terlampaui. Tingkat keuntungan yang diperoleh oleh perusahaan sangat dipengaruhi oleh berbagai faktor antara lain tunggakkan yang kecil, jasa yang diberikan, harga jual yang ditetapkan dan kemampuan sumber daya manusia yang dilakukan mulai dari perusahaan itu berdiri sampai perusahaan masih melakukan produksi. Peningkatan produksi diperoleh berkat adanya kerja keras dengan bagian yang ada dalam perusahaan**.**

Untuk mendapatkan bonus cadangan bad debt ada syarat yang dibuat perusahaan yang harus di penuhi oleh karyawan atau penagih yaitu tunggakkan harus dibawah 5%, apabila tunggakkan diatas 5% maka bonus cadangan bad debt tidak dikeluarkan oleh perusahaan atau tidak dibagi kepada karyawan.

(Dita Febriyani ; 2012 : 2)

II.4. Pengertian MySQL

MySQL *database server* adalah RDBMS (*Relasional Database Management Sistem*) yang dapat menangani data bervolume besar. Meskipun begitu, tidak menuntut *resource* yang besar. MySQL adalah *database* yang paling populer diantara *database* yang lain.

MySQL adalah program database yang mampu mengirim dan menerima data dengan sangat cepat dan multi user. MySQL memiliki dua bentuk lisensi, yaitu free software dan shareware. Penulis sendiri dalam menjelaskan buku ini menggunakan MySQL yang free software karena bebas menggunakan databse ini untuk keperluan probadi atau usaha tanpa harus membeli atau membayar lisensi yang berada di bawah lisensi GNU/GPL (General Public License), yang dapat di download pada alamat resminya *http://www.mysql.com*. MySQL sudah cukup lama dikembangkan, beberapa *fase* penting dalam pengembangan MySQL adalah sebagai berikut:

- MySQL dirilis pertama kali secara secara internal pada 23 Mei 1995
- Versi *windows* dirilis pada 8 Januari 1998 untuk *windows* 95 dan *windows* NT
- Versi 3.23 : beta dari Juni 2000, dan dirilis pada Januari 2001.
- Versi 4.0 : beta dari Agustus 2002, dan dirilis pada Maret 2003 (unions)

(Wahana Komputer ; 2010 ; 5)

II.5. *Unified Modelling Languag* (UML)

UML singkatan dari Unified Modeling Language yang berarti bahasa pe,odelan standar. Chonoles mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantic. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. UML

bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. (Prabowo et al; 2011 : 6)

UML diaplikasikan untuk maksud tertentu, biasanya antara lain:

1. Merancang perangkat lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembangan sistem berorientasi objek menggunakan bahasa model untuk menggambar, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dengan mengaplikasikan beragam sistem. Intinya, UML merupakan alat komunikasi yang konsisten dalam mensupport para pengembangan sistem saat ini. (Prabowo Pudjo Widodo, Herlawati; 2011 : 6-7)

Beberapa leteratur menyebutkan bahwa UML menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain:

1. Diagram Kelas, bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodela sistem berorientasi objek . Meskipun bersifat statis, sering pula diagram kelas mmemuat kelas-kelas aktif.
2. Diagram paket (Package Diagram), bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.
3. Diagram *use case*, bersifat statis. Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram interaksi dan *sequence* (urutan), bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram komunikasi (*Communication Diagram*), bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi UML 14 yang menekankan organisasi structural dari objek-objek yang menerima seta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram*), bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*state*), transisi, kejadian serta aktifitas. Diagram ini teruama penting

untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi, dan terutama penting pada pemodelan sistem-sisten yang reaktif.

7. Diagram aktivitas (*Aktivity Diagram*), bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi sistem dan member tekanan pada aliran kendali antar objek.
8. Diagram komponen (*Component Diagram*), bersifat statis. Diagram komponen ini memeperlihatkan organisasi serta keberuntungan sistem / perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas-kelas, antaarmuka-antramuka serta kolaborasi-kolaborasi.
9. Diagram *Deployment (Deployment Diagram)*, bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*). Memuat simpul-simpul beserta komponen-komponen yang ada di dalamnya. Diagram *deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*Distributed Computing*)

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada UML dimungkinkan kita menggunakan diagram-diagram lainnya (misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya). (Prabowo Pudjo Widodo, Herlawati; 2011 : 10-12)

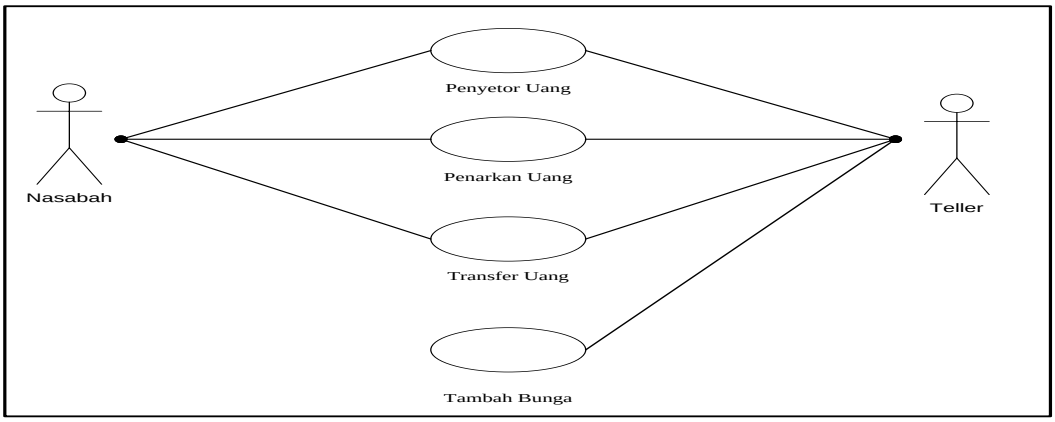
II.5.1. Diagram *Use Case* (*Use Case Diagram*)

Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. MenurutPooley (2005:15) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

Komponen pembentuk diagram *use case* adalah:

1. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use Case*, aktifitas/sarana yang disiapkan oleh bisnis / sistem.
3. Hubungan (*Link*), actor mana saja yang terlibat dalam *use case* ini.

Gambar dibawah ini merupakan salah satu contoh bentuk diagram *use case* yaitu :

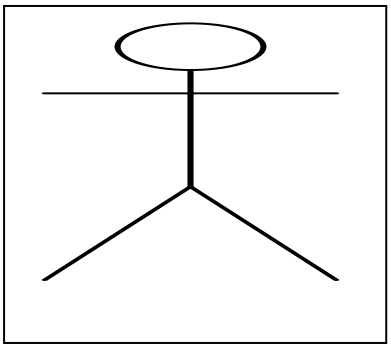


Gambar II.1. Diagram *Use Case*

(Sumber : Prabowo Pudjo Widodo, Herlawati ; 2011 : 17)

1. Aktor

Menurut Chonoles (2003 : 17) menyarankan sebelum membuat *use case* dan menentukan aktornya, agar mengidentifikasikan siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder*.



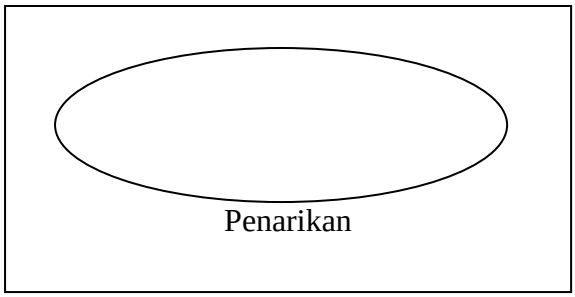
Gambar II.2. Aktor

(Sumber : Prabowo Pudjo Widodo, Herlawati ; 2011 : 17)

2. Use Case

Menurut Pilone (2005 : 21) use case menggambarkan fungsi tertentu dalam suatu sistem berupa komponen, kejadian atau kelas. Sedangkan

menurut Whitten (2004 : 258) mengartikan use case sebagai urutan langkah-langkah yang secara tindakan saling terkait (scenario), baik terotomatisasi maupun secara manual, untuk tujuan melengkapi suatu tugas bisnis tunggal. Use case digambarkan dalam bentuk ellips/oval pada gambar II.3 sebagai berikut :



Gambar II.3. Simbol Use Case

(Sumber : Prabowo Pudjo Widodo, Herlawati ; 2011 : 22)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu (Chonoles, 2003: 22) menawarkan cara untuk mengahasilkan use case yang baik, yakni:

1. Pilihlah nama yang baik

   Use case adalah sebuah *behavior* (perilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detil, tambahkan kata benda yang mengidentifikasikan dampak aksinya terhadap suatu kelas objek. Oleh karena itu diagram *use case* seharusnya berhubungan dengan diagram kelas.

2. Ilustrasikan perilaku dengan lengkap

Use case dimulai dari inisiasi oleh aktor primer dan berakhir pada actor dan menghasilkan tujuan. Jangan membuat *use case* kecuali Anda mengetahui tujuannya. Sebagai contoh, memilih jenis tempat tidur (*king size, queen size* atau dobel) saat tamu memesan tidak dapat dijadikan use case karena merupakan bagian dari use case pemesanan kamar dan tidak dapat berdiri sendiri (tidak mungkin tamu memesan kamar tidur jenis king tapi tidak memesan kamar hotel).

3. Identifikasi perilaku dengan lengkap

Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, use case harus lengkap. Ketika member nama pada *use case*, plihlah frasa kata kerja yang implikasinya hingga selesai. Misalnya gunakan frasa *reserve a room* (pemesanan kamar) dan jangan *reserving a room* (memesan kamar) karena memesan kamar menggambarkan perilaku yang belum selesai.

4. Menyediakan *Use case* lawan (inverse)

Kita biasanya membutuhkan *use case* yang membatalkan tujuan misalnya pada *use case* pemesanan kamar, dibutuhkan pada *use case* pembatalan pesanan kamar.

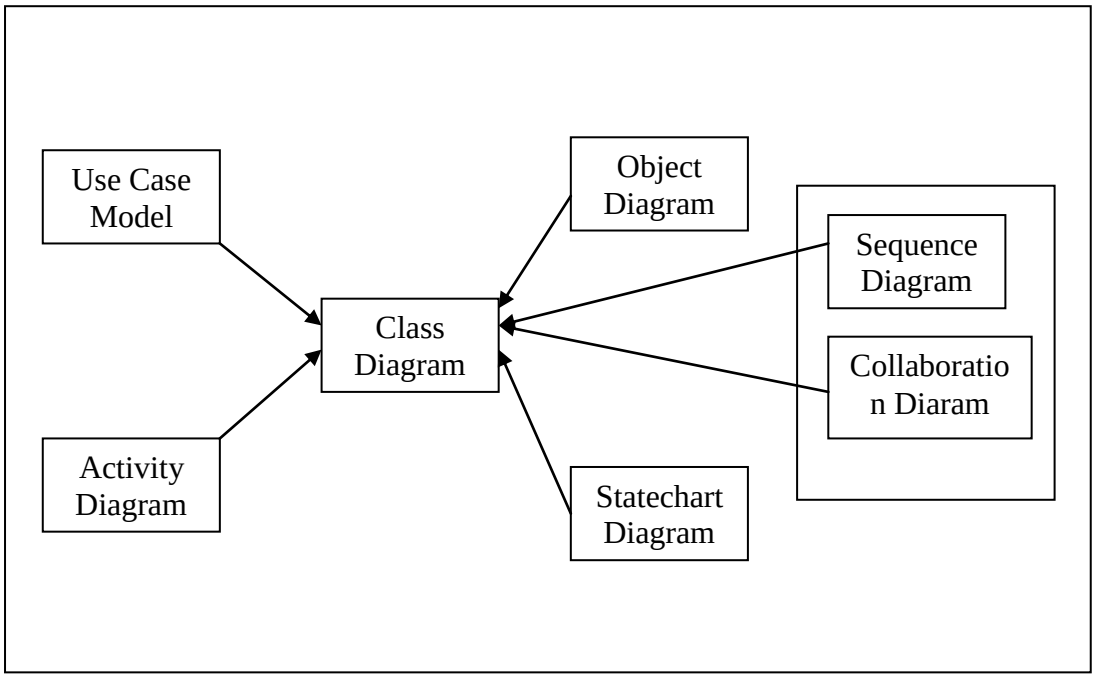
5. Batasi Use case hingga satu perilaku saja

Kadang kita cenderung membuat *use case* yang menghasilkan lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah *use case* kita hanya fokus pada satu hal. Misalnya, pengguna *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda. Untuk

menyediakan penjelasan detail terhadap segala kemungkinan yang terjadi pada *use case*, apa yang terjadi dan apa respon sistem.

II.5.2. Diagram kelas (Class Diagram)

Diagram kelas adalah inti dasri proses pemodelan objek, baik forward engineering maupun reverse engineering memanfaatkan diagram ini. Forward engineering adalah proses perubahan model menjadi kode program dsedangkan reserve engineering sebaliknya merubah kode program menjadi model.



Gambar II.4. Hubungan Diagram Kelas Dengan Diagram UML lainnya

(Sumber : Prabowo Pudjo Widodo, Herlawati ; 2011 : 38)

II.5.3. Diagram Aktivitas (Activity Diagram)

Diagram aktivitas lebih memfokuskan dari pada eksekusi dan alur sistem dari pada bagaimana sistem itu dirakit. Diagram ini tidak hanya memodelkan

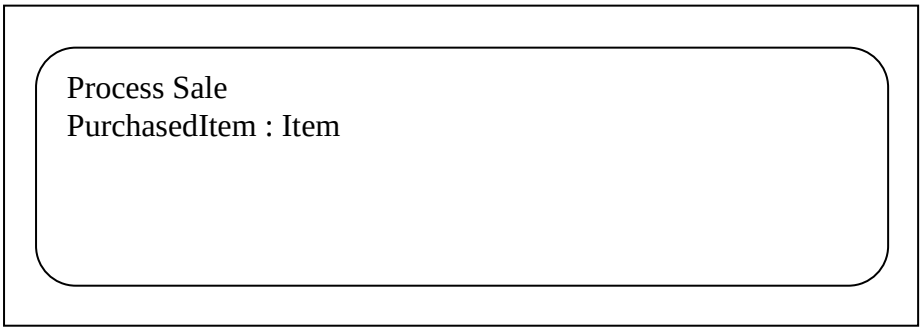
model bisnis juaga. Diagram aktivitas menunjukkan aktivitas sistem dalam bentuk kumpulan aksi-aksi. Ketika digunakan dalam pemodelan software, diagram aktivitas mempresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian di luar seperti pemesanan atau kejadian-kejadian internal misalnya proses penggajian tiap jumat sore.

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi melakukan langkah sekali saja tidak boleh dipecah menjadi beberapa langkah lagi. Contoh akasi yaitu:

1) Fungsi matematika
2) Pemanggilan perilaku
3) Pemrosesan data

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu classifier, classifier dikatakan konteks dari aktivitas. Aktivitas dapat mengakses atribut dan operasi classifier, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan untuk model proses bisnis, informasi itu biasanya disebut process-relevant data. Aktivitas diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya specific dan digunakan hanya untuk aktivitas tertentu.

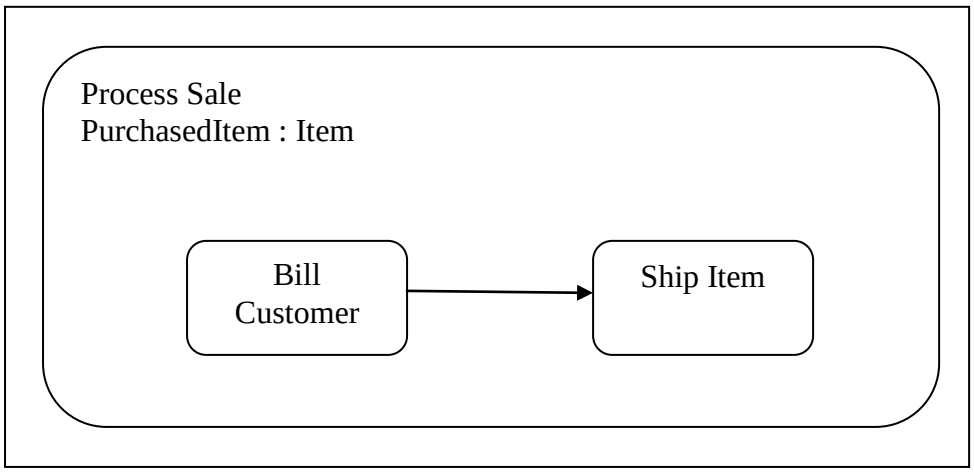
Aktivitas digambarkan dengan persegi panjang tumpu, namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.5. Aktivitas Sederhana Tanpa Rincian

(Sumber : Prabowo Pudjo Widodo, Herlawati ; 2011 : 145)

Detail aktivitas dapat dimasukkan di dalam kotak. Akan diperlihatkan dengan symbol yang sama dengan aktivitas dan namanya diletakkan didalam persegi panjang.



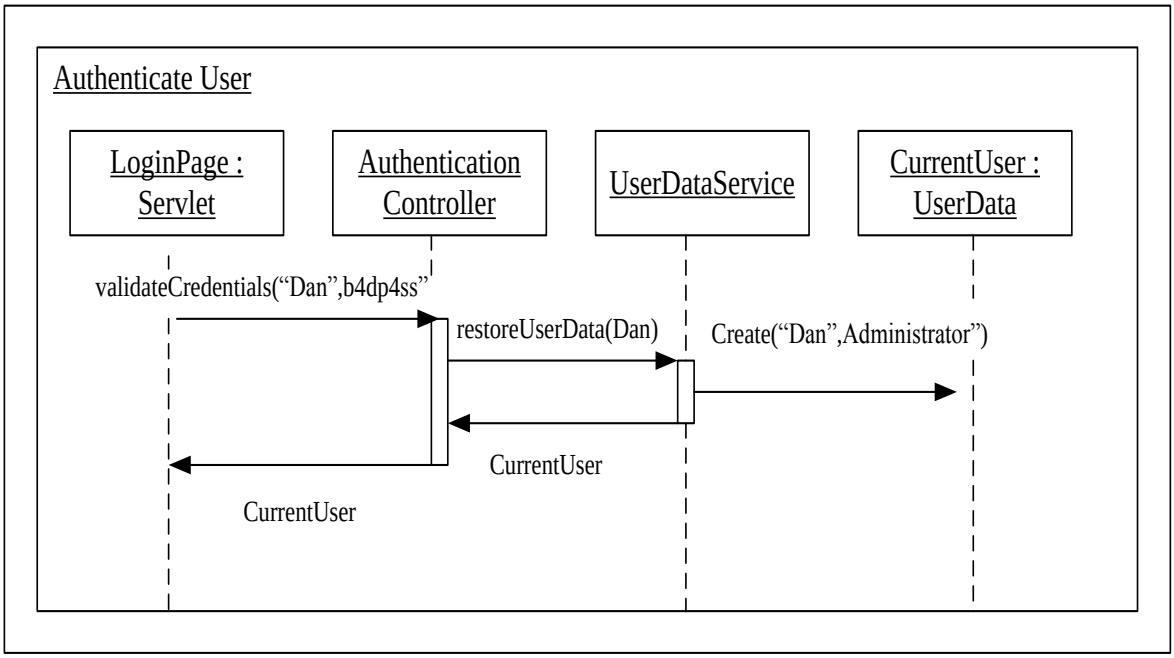
Gambar II.6. Aktivitas Dengan Detail Sederhana

(Sumber :Prabowo Pudjo Widodo, Herlawati ; 2011 : 145)

II.5.4. Sequence Diagram

Menurut (Douglas, 2004 : 174) menyebutkan ada tiga diagram primer UML dalam memodelkan scenario interaksi, yaitu: diagram urutan (*sequence diagram),* diagram waktu (*timing diagram*) dan diagram komunikasi (*communication diagram*). Namun demikian (Pilone, 2005 : 174) menyatakan

bahwa diagram yang paling banyak dipakai adalah diagram urutan. Gambar II.6. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nanti.



Gambar II.7. Contoh Diagram Urutan

(Sumber : Prabowo Pudjo Widodo, Herlawati ; 2011 : 175)

II.6. ERD (*Entity Relationship Diagram*)

Menurut Wahana Komputer; 2010:30. Pada dasarnya ERD (*Entity Relationship Diagram*) adalah sebuah diagram yang secara konseptual memetakan hubungan antar penyimpanan pada diagram DFD di atas. ERD ini digunakan untuk melakukan permodelan terhadap struktur data dan hubungannya. Penggunaan ERD ini dilakukan untuk mengurangi tingkat kerumitan penyusunan sebuah database yang baik.

Entity dapat berarti sebuah obyek yang dapat dibedakan dengan obyek lainnya. Obyek tersebut dapat memiliki komponen – komponen data (atribut atau field) yang membuatnya dapat dibedakan dari obyek yang lain. Dalam dunia database entity memiliki atribut yang menjelaskan karakteristik dari entity tersebut. Ada dua macam atribut yang dikenal dalam entity yaitu atribut yang berperan sebagai kunci primer dan atribut deskriptif. Hal ini berarti setiap entity memiliki himpunan yang diperlukan sebuah primary key untuk membedakan anggota – anggota dalam himpunan tersebut.

Atribut dapat memiliki sifat – sifat sebagai berikut :

- Atomic, atomik adalah sifat dari atribut yang menggambarkan bahwa atribut tersebut berisi nilai yang spesifik dan tidak dapat dipecah lagi. Contoh dari sifat atomik adalah field status dari tabel karyawan yang hanya berisi menikah atau single.
- multivalued, sifat ini menandakan atribut ini bisa memiliki lebih dari satu nilai untuk tiap entity tertentu. Misalnya adalah field hobi, hobi dari tiap karyawan mungkin dan hampir pasti lebih dari satu. Misalnya karyawan A memiliki hobi : membaca, nonton TV dan bersepeda.
- Composite, atribut yang bersifat komposit adalah atribut yang nilainya adalah gabungan dari beberapa atribut yang bersifat atomik. Contohnya adalah atribut alamat yang dapat dipecah menjadi atribut atomik berupa alamat, kode pos, no telepon, dan kota.

ERD (*Entity Relationship Diagram*) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut

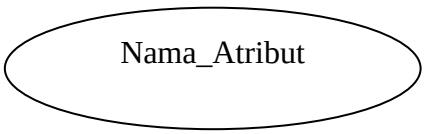
entitas dan hubungan antar objek. Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain.

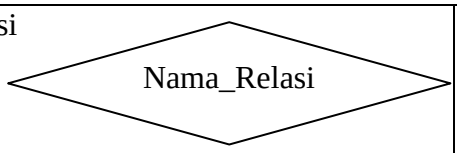
Sebagai tambahan, model ER menyajikan pula batasan dimana isi basis data harus menyesuaikan dengan batasan. Salah satu batasan yang penting adalah pemetaan kardinalitas (*mapping cardinalities*), yang menggambarkan jumlah entitas yang berhubungan dengan entitas lain melalui suatu relasi. (Janner Si ; 2010 : 60-61)

II.6.1. Simbol-simbol ERD (*Entity Relationship Diagram*)

Adapun symbol-simbol ERD (*Entity Relationship Diagram*) ditunjukkan pada table II.3.

Tabel II.1. Simbol-Simbol ERD (*Entity Relationship Diagram*)

Simbol	**Deskripsi**
Entitas / *entity* 	Entitas merupakan data inti yang akan disimpan; bakal table pada basis data
Atribut 	Fiel atau koilom data yang perlu disimpan dalam suatu entitas

Relasi 	Relasi yang menghubungkan antara entitas; relasi biasanya diawali dengan kata kerja
Asosiasi / *Association* **1 0..***	Penghubung antar relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian

Sumber : (Janner Si ; 2010 : 59-60)

II.7. Normalisai

Normalisasi adalah bagian perancangan basisdata. Tanpa Normalisasi, sistem basisdata menjadi tidak akurat, lambat, tidak efisien, serta tidak memberikan data yang diharapkan.

Pada waktu menormalisasi basisdata, ada empat tujuan yang harus dicapai, yaitu:

1. Mengatur data dalam kelompok-kelompok sehingga masing-masing kelompok hanya menangani bagian kecil sistem.
2. Meminimal jumlah data berulang dalam basisdata.
3. Membuat basisdata yang datanya diakses dan manipulasi secara cepat dan efisien tanpa melupakan integritas data.
4. Mengatur data sedemikian rupa sehingga ketika memodifikasi data, hanya mengubah satu tempat.

Tujuan Normalisai adalah membuat kumpulan table relasional yang bebas dari sata berulang dan dapat dimodifikasi secara benar dan konsisten. Ini berarti bahwa semua table pada basisdata relasional harus berada pada bentuk normal ketiga (3NF). Sebuah table relasional berada pada 3NF jika dan hanya jika semua kolom bukan kunci adalah saling independen berarti bahwa tidak ada kolom bukan kunci yang tergantung pada sembarang kombinasi kolom lainnya. Dua bentuk normal pertama aadalah langkah antara untuk mencapai tujuan, yaitu mempunyai semua table dalam 3NF. (Janner Si; 2010 : 77-78)

Tahapan normalisai terdiri dari beberapa bentuk yaitu sebagai berikut:

1. Bentuk normal pertama (1NF / *First Normal Form*)

   Bentuk normal pertama memiliki ciri yaitu data berbentuk *flat file* (file datar), *record* disusun sesuai kedatangan, masih mungkin terjadi penyimpangan data (*anomaly* data). *Anomali* data dapat berupa *insert anomaly, delete anomaly*, *update anomaly* dan *redundancy* data (data duplikat). (Janner Si; 2010 : 79)

2. Bentuk Normal Kedua (2NF/ *Second Normal Form*)

   Bentuk normal kedua memiliki ciri yaitu tidak terjadi *anomali* data, setiap *field / attribute* bukan kunci harus tergantung fungsi (*functional dependency*) terhadap *field / attribute* kunci, masih mungkin terjadi *transitive dependency* (*field* bukan kunci tergantung pada *field* bukan kunci dalam satu table). (Janner Si; 2010 : 81)

3. Bentuk Normal Ketiga (3NF / *Third Normal Form*)

Bentuk normal ketiga memiliki syarat yaitu table harus tidak terdapat transitive dependency (*field* bukan kunci tergantung pada *field* bukan kunci dalam satu tabel). (Janner Si; 2010 : 82)

4. Bentuk Normal *Boyce Codd* (BCNF / *Boyce Codd Normal Form*)

Pada tahap ini menghilangkan ketergantungan *field* bukan kunci secara persial (bagian) kunci dalam satu tabel. Apabila pada normal ketiga tidak lagi ditemukan *field* bukan kunci tergantung secara persial dalam satu tabel, maka normal ketiga juga merupakan bentuk BCNF. (Janner Si; 2010 : 84)

5. Bentuk Normal Kelima (5NF / *Five Normal Form*)

Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan *(join dependence*). Ketergantungan gabungan berarti bahwa sebuah tabel., setelah didekomposisi menjadi tiga atau lebih tabel yang lebih kecil, harus dapat digabungkan kembali untuk membentuk tabel asal. Dengan kata lain, 5NF menunjukkan ketika sebuah tabel tidak dapat didekomposisi lagi. (Janner Si ; 2010 : 85)

II.8. Mengenal Visual Basic

Menurut Edy Winarto; 2010:1. Visual Basic dibuat oleh microsoft, merupakan salah satu bahasa pemrograman berorientasi objek yang mudah dipelajari. Selain menawarkan kemudahan, Visual Basic juga cukup andal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi database. Visual basic merupakan bahasa pemrograman event drive, dimana program akan

menunggu sampai ada respons dari user/pemakai program aplikasi yang dapat berupa kejadian atau event, misalnya ketika user mengklik tombol atau menekan enter. Jika kita membuat aplikasi dengan visual basic maka kita akan mendapatkan file yang menyusun aplikasi tersebut, yaitu :

1. File Project (*.vbp)

   File ini merupakan kumpulan dari aplikasi yang kita buat. File project bisa berupa file *.frm, *.dsr atau file lainnya.

2. File Form (*.frm)

   File ini merupakan file yang berfungsi untuk menyimpan informasi tentang bentuk form maupun interface yang kita buat.